1. Program Development

Language Used: Java
Package: org.example

Program Description:
This Java program evaluates student performance by:

1) Adding student records with name and marks.
2) Calculating class average.
3) Assigning grades (A/B/C/Fail).
4) Handling invalid input scenarios.

Source Code:

```
package org.example;

import java.util.ArrayList;
import java.util.List;

public class StudentEvaluator {
    static class Student {
        String name;
        int marks;

        public Student(String name, int marks) {
            if (name == null || name.trim().isEmpty()) {
                throw new IllegalArgumentException("Name cannot be empty");
            }
            if (marks < 0 || marks > 100) {
                throw new IllegalArgumentException("Marks must be between 0 and 100");
            }
            this.name = name;
            this.marks = marks;
        }
    }

    private final List<Student> students = new ArrayList<>();

    public void addStudent(String name, int marks) {
        students.add(new Student(name, marks));
    }

    public double calculateAverage() {
        if (students.isEmpty()) return 0.0;
        int total = 0;
        for (Student s : students) {
            total += s.marks;
        }
        return (double) total / students.size();
    }

    public String evaluateGrade(int marks) {
        if (marks >= 90) return "A";
        else if (marks >= 75) return "B";
        else if (marks >= 50) return "C";
        else return "Fail";
    }

    public int getStudentCount() {
        return students.size();
    }
}
```

2. Write Partial Unit Tests

a) Functions Tested and Why:

- addStudent() and getStudentCount() to verify proper addition.
- calculateAverage() to ensure correct average calculation.
- evaluateGrade() to test different grading thresholds.

b) Test Code

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;


public class StudentEvaluatorTest {

   StudentEvaluator evaluator = new StudentEvaluator();


   @Test

   void testAddStudentAndCount() {

       evaluator.addStudent("Alice", 85);

       assertEquals(1, evaluator.getStudentCount());

   }
```

```
    @Test
    void testCalculateAverage() {
        evaluator.addStudent("Bob", 80);
        evaluator.addStudent("Charlie", 70);
        assertEquals(75.0, evaluator.calculateAverage());
    }

    @Test
    void testEvaluateGrade() {
        assertEquals("A", evaluator.evaluateGrade(95));
        assertEquals("B", evaluator.evaluateGrade(78));
        assertEquals("C", evaluator.evaluateGrade(55));
        assertEquals("Fail", evaluator.evaluateGrade(40));
    }
}
```

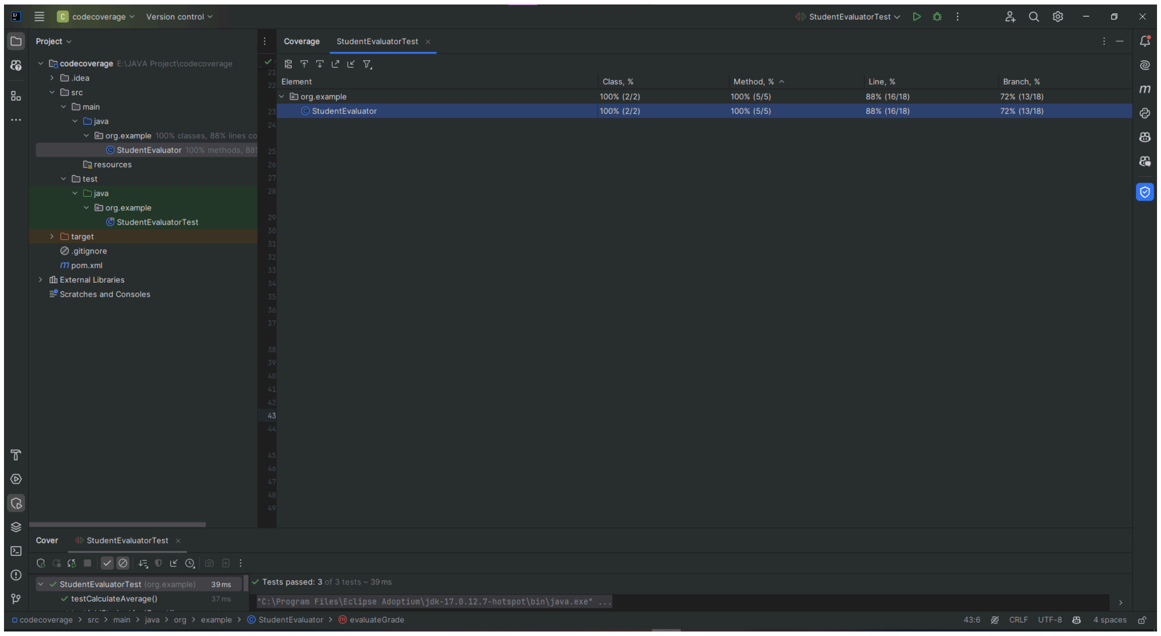
3. Measure Code Coverage

a) Tool Used: JaCoCo with IntelliJ IDEA

b) Installation/Execution:
- Built-in in IntelliJ
- Right-click test class → Run with Coverage

c) Coverage Results Before Extra Tests:
- Class: 100%
- Method: 100%
- Line: 88%
- Branch: 72%

d) Screenshot:



Element	Class, %	Method, %	Line, %	Branch, %
org.example	100% (2/2)	100% (5/5)	88% (16/18)	72% (13/18)
StudentEvaluator	100% (2/2)	100% (5/5)	88% (16/18)	72% (13/18)

```
StudentEvaluatorTest (org.example)   39 ms
  testCalculateAverage()             37 ms
Tests passed: 3 of 3 tests – 39 ms
"C:\Program Files\Eclipse Adoptium\jdk-17.0.12.7-hotspot\bin\java.exe" ...
```

4. Improve Coverage

a) Additional Cases Tested:

- Empty student list
- Invalid inputs: empty name, negative marks, marks > 100

b) Improved Test Cases:

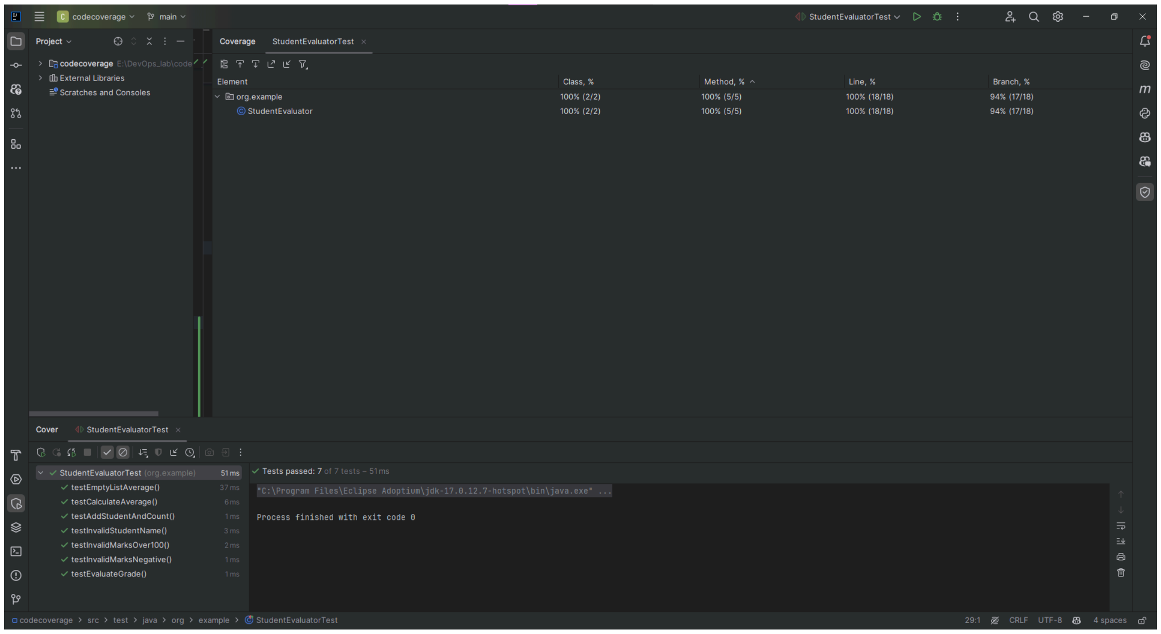
```java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;


public class StudentEvaluatorTest {

    StudentEvaluator evaluator = new StudentEvaluator();


    @Test

    void testAddStudentAndCount() {

        evaluator.addStudent("Alice", 85);

        assertEquals(1, evaluator.getStudentCount());

    }
```

```
   @Test
   void testCalculateAverage() {
       evaluator.addStudent("Bob", 80);
       evaluator.addStudent("Charlie", 70);
       assertEquals(75.0, evaluator.calculateAverage());
   }

   @Test
   void testEvaluateGrade() {
       assertEquals("A", evaluator.evaluateGrade(95));
       assertEquals("B", evaluator.evaluateGrade(78));
       assertEquals("C", evaluator.evaluateGrade(55));
       assertEquals("Fail", evaluator.evaluateGrade(40));
   }

   @Test
   void testInvalidStudentName() {
       assertThrows(IllegalArgumentException.class, () ->
evaluator.addStudent("", 90));
   }

   @Test
   void testInvalidMarksNegative() {
       assertThrows(IllegalArgumentException.class, () ->
evaluator.addStudent("John", -5));
   }

   @Test
```

```
    void testInvalidMarksOver100() {

        assertThrows(IllegalArgumentException.class, () ->
evaluator.addStudent("John", 105));

    }



    @Test

    void testEmptyListAverage() {

        assertEquals(0.0, new StudentEvaluator().calculateAverage());

    }

}
```

c) Updated Coverage:
- Class: 100%
- Method: 100%
- Line: 100%
- Branch: 9%

d) Screenshot:



Element	Class, %	Method, %	Line, %	Branch, %
org.example	100% (2/2)	100% (5/5)	100% (18/18)	94% (17/18)
StudentEvaluator	100% (2/2)	100% (5/5)	100% (18/18)	94% (17/18)

✓ StudentEvaluatorTest (org.example)	51 ms
✓ testEmptyListAverage()	37 ms
✓ testCalculateAverage()	6 ms
✓ testAddStudentAndCount()	1 ms
✓ testInvalidStudentName()	3 ms
✓ testInvalidMarksOver100()	2 ms
✓ testInvalidMarksNegative()	1 ms
✓ testEvaluateGrade()	1 ms

✓ Tests passed: 7 of 7 tests – 51 ms

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.12.7-hotspot\bin\java.exe" ...

Process finished with exit code 0
```

codecoverage › src › test › java › org › example › StudentEvaluatorTest